HAL
open science

Programmation tangible pour les enfants : analyse de l'existant, classification et opportunités

Julie Henry, Bruno Dumas, Antoine Bodart

To cite this version:

Julie Henry, Bruno Dumas, Antoine Bodart. Programmation tangible pour les enfants : analyse de l'existant, classification et opportunités. 30eme conférence francophone sur l'interaction homme-machine, Oct 2018, Brest, France. 9p. hal-01899246

HAL Id: hal-01899246

https://hal.science/hal-01899246v1

Submitted on 19 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Programmation tangible pour les enfants : analyse de l'existant, classification et opportunités

Julie Henry
PReCISE, NADI, UNamur
Namur, Belgique
julie.henry@unamur.be

Bruno Dumas
PReCISE, NADI, UNamur
Namur, Belgique
bruno.dumas@unamur.be

Antoine Bodart
PReCISE, NADI, UNamur
Namur, Belgique

Résumé

Cet article présente une étude des dispositifs tangibles soutenant les enfants (de 2 à 12 ans) dans leur apprentissage des concepts de base de la programmation. Vingt dispositifs sont examinés à travers deux grilles de lecture : la taxonomie de Fishkin et une classification originale proposée par les auteurs. Cette dernière s'intéresse à deux aspects en particulier : la didactique, en essayant d'identifier les concepts mis en œuvre par l'enfant, et les modalités d'interaction mises à la disposition de cet enfant pour communiquer et interagir avec le dispositif. Des pistes d'innovation en terme de design d'interaction sont suggérées et des opportunités de continuer cette étude, notamment en ce qui concerne l'assimilation des concepts par l'enfant, sont présentées.

Mots Clés

Tangible ; Enseignement de la programmation ; Apprentissage de la programmation ; Enfant ; Classification des dispositifs ; Taxonomie de Fishkin

Abstract

This article presents a study of the tangible devices supporting children (aged 2 to 12) in their learning of basic programming concepts. Twenty devices are examined through two perspectives: Fishkin's taxonomy and an original classification from the authors. The latter focuses on two aspects:

Dispositifs
Physiques, avec de l'électronique
Bee-Bot [13]
Dr. Wagon [1]
E-Block [37, 36]
Electronic Blocks [39, 38]
KIBO [29, 2]
Robo-Blocks [26]
Hybrides, avec des blocs de programmation tangibles
Algoblock [30]
Dialando [27]
Pleo [21]
Quetzal [6, 7]
StarLoop [14]
Strawbies [11]
Tangicons [24]
TanPro-Kit [33]
T-Butterfly [22, 23]
Tern [7, 8, 10]
T-Maze [35, 34]
Toque [32]
T-ProRob [22, 23]
TurTan [4]

Tableau 1: Organisation des dispositifs analysés selon leurs caractéristiques physiques

didactics, by trying to identify the concepts implemented by the child, and the interaction modalities made available to a child to communicate and interact with the device. Innovation tracks in terms of interaction design are suggested and opportunities to continue this study, particularly with regard to the assimilation of concepts by the child, are presented.

Author Keywords
Tangible, ; TUI ; Programming Education ; Children ; Fishkin's Taxonomy ; Classification of Learning Aid Tools

CCS Concepts
Social and professional topics → Computing education programs → *Computer science education*

Introduction
Depuis des décennies, l'enseignement de la programmation à des novices fait l'objet de nombreuses recherches autour de thématiques aussi variées que le curriculum et la didactique associée (quoi enseigner), la pédagogie (comment l'enseigner) et les dispositifs d'aide à l'apprentissage (avec quoi). Ces derniers sont de plus en plus nombreux pour venir en aide aux enfants dans l'exploration des concepts de base de la programmation et de la pensée informatique [12, 5, 16, 28, 41]. Cependant, malgré cette diversité de l'offre, l'apprentissage de la programmation reste difficile [19, 18].

L'interaction tangible est une approche privilégiée pour aider les enfants à développer des compétences en résolution de problèmes [25]. Des travaux de recherche pionniers en la matière, concernant entre autres l'apprentissage de la programmation à l'aide de robots ou de dispositifs embarqués, peuvent être attribués à Papert [15] et Perlman [17]. Cependant, la réponse apportée par le tangible au défi d'enseigner la programmation reste partiellement couverte. En effet, si certains s'intéressent depuis peu à la compréhension "profonde" qu'ont les enfants des concepts-clé en programmation [31], beaucoup restent à évoquer la simple performance (tâche réussie) [23], ou encore des aspects tels que l'engagement, le plaisir, la motivation et la collaboration comme résultats observés suite à l'utilisation d'un dispositif d'aide à l'apprentissage de la programmation [40, 9]. On peut également regretter le caractère exclusif qu'occupent bien souvent les dispositifs dans de nombreuses études de cas : un outil unique aidant à l'apprentissage de l'ensemble des concepts de base en programmation. Et si l'apprentissage de chaque concept pouvait être optimisé à travers un outil qui lui serait particulièrement bien adapté ?

Dans cet article, nous présentons les résultats d'une étude de l'existant en ce qui concerne les dispositifs tangibles dédiés à l'apprentissage de la programmation. Nous avons sélectionné des dispositifs ayant essentiellement fait l'objet de recherches académiques et les avons examinés à travers deux taxonomies : celle de Fishkin [3] et une taxonomie originale développée dans le cadre de cette étude. Partant de là, nous tirons les premières conclusions et discutons les potentielles suites à cette étude.

Méthodes et questionnement
Les dispositifs tangibles sont particulièrement présents lorsqu'il s'agit d'initier des enfants à la programmation. Pourtant, il existe peu de retours sur les réels apports de tels dispositifs en terme d'apprentissages. À travers une étude de l'existant, nous espérons identifier les qualités et les manques des dispositifs actuels et proposer des pistes de solution aux successeurs de ceux-ci.

Vingt dispositifs ont été décortiqués à la lumière de critères portant sur des aspects didactiques, mais également sur

Incarnation

Complète : L'*output* est l'*input*, à savoir que l'action mise en œuvre via un objet produit un résultat visible sur ce même élément.

Proche : L'*output* se produit à proximité de l'*input*.

Environnement : Il s'agit souvent d'effets (audio, lumière, etc.) produits autour de l'utilisateur.

Distante : Le résultat s'affiche sur un objet qui pourrait très bien se trouver géographiquement éloigné de l'*input*.

Métaphore

Nom : La forme de l'*input* est similaire à la forme (physique, sonore, etc.) de l'objet réel qu'il représente, indépendamment de l'action de l'utilisateur.

Verbe : Elle caractérise une analogie avec l'action qui est réalisée, indépendamment de l'objet avec laquelle l'utilisateur la réalise

Complète : Une analogie n'est plus utile ; l'utilisateur manipule un objet et le monde change en fonction.

des aspects spécifiques au domaine des Interfaces Homme-Machine. Ces critères pourraient compléter ceux proposés à travers d'autres études, notamment les perspectives évoquées par Yu et Roque [41].

Sélection des dispositifs
La sélection des dispositifs mis à l'étude s'est basée sur la recherche d'articles scientifiques via Google Scholar, en utilisant les mots-clé "Tangible", "Programming" et "Children". Les articles choisis étaient tous cités dans les dix premières pages de résultat. Sur base d'une lecture rapide, seuls les articles faisant mention d'un dispositif tangible clairement développé pour l'enseignement de la programmation ont été retenus. Cette même recherche a également été effectuée en français, ajoutant un article au corpus. Quelques articles supplémentaires ont été retenus par lecture des références citées dans le corpus de base.

En s'inspirant de l'organisation basée sur les caractéristiques physiques proposée par Yu et Roque [41], les dispositifs sélectionnés peuvent être classés en deux groupes (voir tableau 1) : physique ou hybride.
Les dispositifs physiques ne comprennent que des éléments tangibles, à l'instar de KIBO [29, 2] qui consiste en un robot et un ensemble de blocs de programmation. À noter que le mot "bloc" est ici à prendre au premier sens du terme, à savoir qu'il s'agit d'une masse constituée d'un seul morceau. Il peut tantôt représenter une pièce contenant de l'électronique, tantôt un cube en bois, tantôt un bout de papier selon le dispositif décrit.
Les dispositifs hybrides comprennent à la fois des éléments physiques et des éléments virtuels. Dans le cas de Strawbies [11], l'élément virtuel consiste en une application développée pour une utilisation sur tablette. Petite particularité, le dispositif Toque [32] n'utilise pas de blocs, mais des accessoires représentant les actions-type en cuisine (couper, mélanger, etc.), associés à un langage de programmation iconique (langage virtuel).

Classification des dispositifs selon Fishkin

	Métaphore			
Incarnation	*Nom*	*Verbe*	*Nom & Verbe*	*Complète*
Complète		Electronic Blocks Pleo	Bee-Bot	
Proche		Electronic Blocks KIBO, StarLoop TurTan		
Environnement				
Distante		Algoblock, Dialando Dr. Wagon, E-Block KIBO, Pleo, Quetzal Robo-Blocks Strawbies Tangicons Tan-Pro Kit T-Butterfly, Tern T-Maze, T-ProRob Toque		

Tableau 2: Classification des dispositifs étudiés selon la taxonomie de Fishkin [3]

Selon Fishkin [3], chaque dispositif peut être défini selon deux axes (voir tableau 2) : l'incarnation et la métaphore. L'incarnation se base sur le degré selon lequel *input* et *output* sont distincts l'un de l'autre, alors que la métaphore juge de l'analogie entre l'effet d'une action de l'utilisateur sur le dispositif et l'effet produit par une action similaire dans le monde réel (voir les détails dans le cadre ci-contre).

Interactions en entrée

Interactions par mouvement Un dispositif est continu s'il récupère un signal continu représentant la position d'un pointeur (un écran tactile, par exemple). Il est discret lorsque l'information récupérée est une valeur booléenne (un bouton est appuyé ou non). Les interfaces embarquées utilisent le dispositif lui-même comme *input*, au moyen de divers capteurs intégrés (de déplacement, inertiels, d'identification ou encore des cellules photosensibles).

Interactions vocales Via des microphones.

Interactions passives L'utilisation de capteurs, embarqués et externes, permettent aux dispositifs de s'adapter à l'environnement, sans action explicite de l'utilisateur.

Interactions en sortie

Soit visuelles, soit auditives, soit kinesthésiques (par retour tactile, comme un vibreur, ou par retour de force).

Classification originale des dispositifs
Outre l'"âge" permettant d'identifier le public-cible de chaque dispositif, la classification originale proposée ici se base sur des critères apportant des éléments de réponses sur les aspects didactiques qui animent (ou pourraient animer) un dispositif et sur les techniques d'interactions privilégiées (voir figure 1).
Au niveau didactique, une distinction est faite entre les concepts de base en programmation mis en œuvre par l'enfant et ceux qu'il pourrait percevoir (sans aller jusqu'à dire "comprendre") lors de l'utilisation du dispositif. Dans un premier temps, seuls les premiers sont envisagés. Ils sont complémentaires aux concepts décrits dans l'article de Yu et Roque [41] comme étant supportés par les dispositifs analysés. De notre point de vue, la distinction didactique est ici encore existante, à savoir que certains concepts plus abstraits tels que la variable peuvent effectivement être supportés sans être pour autant perçus par l'enfant.
Pour ce qui est du design d'interaction, le dispositif est notamment décrit à travers ses *inputs* et ses *outputs* (voir l'espace de classification de Roudaut et Lecolinet [20] dans le cadre ci-contre).

Analyse et discussion

L'ensemble des dispositifs étudiés incluant un robot sont des dispositifs physiques (voir tableau 1). Par définition, tous les dispositifs décrits comprennent de l'électronique. Cependant, celle-ci n'est pas toujours située au niveau des blocs de programmation. Ainsi, le dispositif physique KIBO [29, 2] dispose de blocs exclusivement composés de bois. Du côté des dispositifs hybrides, Quetzal [6, 7], Tangicons [24], Tern [7, 8, 10] et T-Maze [35, 34] proposent également des blocs exempt d'électronique. Dans le contexte de dispositifs tangibles, cette organisation pourrait être augmentée par une description plus poussée des blocs de programmation. La présence d'électronique ou non dans ceux-ci est un exemple parmi d'autres. On pourrait également évoquer l'existence de "contraintes syntaxiques" posées en vue d'aider l'enfant dans sa construction, à l'instar des dispositifs Quetzal [6, 7], Strawbies [11] et Tern [7, 8, 10] qui jouent sur la forme de blocs, ou encore T-Maze [35, 34] qui utilise les propriétés physiques des matériaux utilisés (aimants). Un dernier exemple pourrait être la représentation physique de certains concepts de base. Ces mêmes dispositifs (excepté T-Maze) et Dr. Wagon [1] ont proposé une telle approche pour le concept de boucle à travers un bloc "englobant", en réalisant physiquement une boucle avec les blocs, ou à l'aide d'un bloc "extensible".

La métaphore de verbe est la plus représentative des dispositifs sélectionnés pour l'étude (voir tableau 2). Les blocs de programmation sont généralement associés à des actions : avancer, tourner à droite, allumer une LED, produire un son, actionner un moteur, etc. Seul dispositif décrit comme nom et verbe, Bee-Bot [13] réalise lui-même toute action programmée à travers l'interface qu'il porte sur son dos. La majorité des dispositifs ont une incarnation distante. Sur les seize classés comme tels, quatorze sont des dispositifs hybrides. Les deux "exceptions" sont des dispositifs centrés sur l'utilisation d'une table interactive : StarLoop [22, 23] et TurTan [4]. On parle donc d'incarnation proche. Trois dispositifs remettent en question le côté exclusif de la taxonomie de Fishkin [3] en se caractérisant par deux incarnations différentes. Pour KIBO [29, 2], l'incarnation proche se justifie par le fait que le robot est nécessaire pour numériser les blocs de programmation et impose donc une certaine proximité. Une fois enregistré, le programme peut être exécuté à distance et mettre en mouvement le robot (incarnation distante). Pleo [21] est une incarnation complète et distante à la fois. C'est la combinaison entre la programmation physique (à travers le robot) et la program-

0. Non décrit dans l'article
A. Âge
1. De 2 à 6 ans
2. De 7 à 12 ans
B. Concepts de programmation mis en œuvre
1. Algorithme/Programme (sans boucle infinie)
2. Séquence
3. Structure conditionnelle
4. Boucle
5. Variable
6. Sous-programme (procédure, fonction)
C. Dispositif physique
1. Tout-en-un
2. Composé
3. Input
a. Continu
b. Discret
c. Capteurs embarqués (mouvement)
d. Caméra vidéo
e. Microphone
f. Autre capteurs (environnement)
4. Output
a. LED, écran, robot (déplacement)
b. Haut-parleurs
c. Dispositifs haptiques

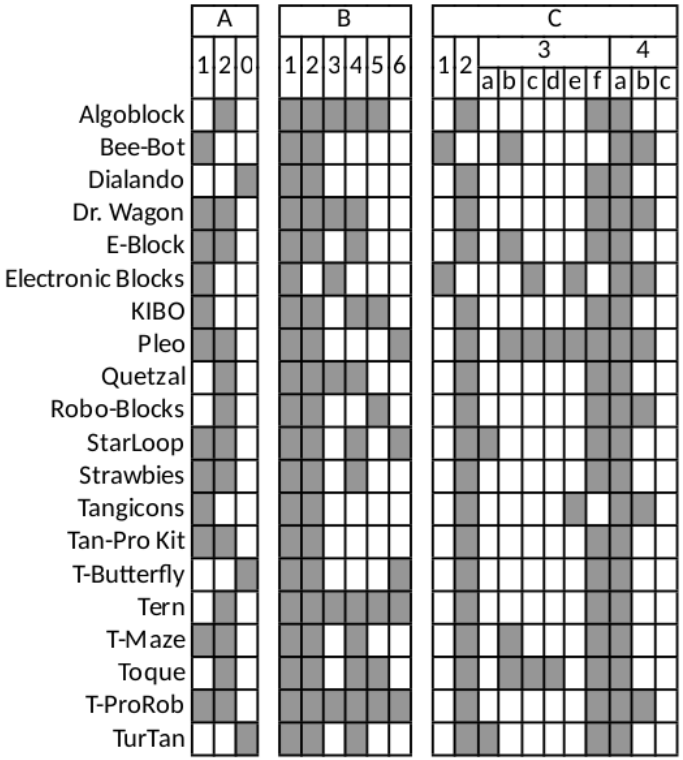

Figure 1: Classification des dispositifs selon les auteurs (en gris, les cases cochées - le critère 0 n'est mentionné que dans le cas où il est utile)

mation à travers un environnement virtuel qui explique ce choix. Enfin, Electronic Blocks [39, 38] se retrouve dans les catégories complète et proche. Si on considère l'ensemble des blocs assemblés comme un dispositif unique, *input* et *output* sont confondus. Par contre, si on regarde en détails chaque bloc, on constate que certains constituent des *inputs* qui provoquent une réaction chez les blocs qui leur sont connectés (et donc proches). Parmi les dispositifs étudiés, Bee-Bot [13] est le seul à présenter une incarnation complète. Dans le dispositif T-Butterfly [22, 23], le parcours formé par les blocs de programmation est exactement le parcours qui sera suivi à l'écran par le caractère (un papillon). Cette caractéristique n'est pas représentable en tant que telle dans la taxonomie de Fishkin [3], bien que pédagogiquement intéressante. Il semble difficile d'aborder la notion de programme informatique sans mettre en œuvre la notion de séquence. Sur les vingt dispositifs étudiés, seulement six, majoritairement (5) destinés à des enfants de 7 ans et plus, considèrent le concept de structure conditionnelle ; douze (dont dix pour les plus de 7 ans), le concept de boucle ; six, les variables et cinq, les sous-fonctions (voir figure 1). Yu et Roque [41] identifient la notion de "donnée" plutôt que de parler de variable. Il serait intéressant de voir si la prise en compte et la différentiation en termes de représentation des rôles de la variable aurait un intérêt.

En ce qui concerne les interactions entre l'enfant et le dispositif, deux dispositifs continus sont rapportés, tous deux mettant en jeu une table interactive (StarLoop [22, 23] et TurTan [4]). Cinq dispositifs discrets sont décrits. Trois présentent des boutons tangibles (Bee-Bot [13], E-Block [37, 36] et T-Maze [35, 34]). Deux proposent une interaction à travers un écran : un met à la disposition de l'enfant un écran tactile (Pleo [21]) et l'autre utilise une combinaison "dispositif de pointage-bouton virtuel" (Toque [32]). Outre ces exceptions, l'ensemble des écrans décrits constituent des modalités de sortie visuelle. La majorité des dispositifs étudiés sont composés. Les caractéristiques liées à l'aspect interactif est "mesuré" pour l'ensemble des éléments. Toutefois, pour ce qui est de l'interaction embarquée, il a été décidé de considérer l'élément central, à

savoir ici les blocs de programmation (ou apparentés). Par capteurs internes, il est considéré des capteurs capables de collecter des données et non simplement de transmettre des données pré-enregistrées. Dans ce dernier cas, à savoir si les blocs renferment en leur sein un système capable de transmettre des données (un identifiant, par exemple), le dispositif dans sa globalité est considéré comme ayant uniquement des capteurs externes. De ce fait, seuls trois dispositifs sont considérés comme ayant des interactions embarquées : Electronic Blocks [39, 38], Pleo [21] et Toque [32].
Mis à part quelques exceptions et selon les conventions prises dans cet article, on est majoritairement face à des techniques d'interaction passives en entrée. Pour ce qui est des modalités de sortie, on reste sur les classiques modalités visuelle et auditive.

Limitations

Notre sélection de dispositifs n'est pas exhaustive, notamment de par sa limitation aux dispositifs ayant fait l'objet d'un article scientifique. L'étude effectuée est, de plus, principalement basée sur les descriptions faites à travers ces articles, soit par les développeurs du dispositif, soit par des chercheurs externes.

Conclusions intermédiaires et travaux futurs

Vingt dispositifs tangibles aidant à l'apprentissage de la programmation ont été sélectionnés et examinés à la lumière de critères issus de trois classifications. Cette étude de l'existant montre qu'il reste encore des opportunités dans ce domaine.

La classification selon la taxonomie de Fishkin souligne des possibles innovations en terme de design d'interaction. Les futurs dispositifs devraient ainsi s'inspirer d'incarnations et de métaphores encore peu (voire pas du tout) représentées. On rejoint et on affine ici les discussions de Yu et Roque quand à la forme que devrait prendre les blocs de programmation. Cette forme pourrait mette en évidence chaque concept, ainsi que les propriétés qui lui sont liées.

La classification originale met en avant, quant à elle, certaines faiblesses, notamment le manque de diversité en ce qui concerne les techniques d'interactions. Au niveau didactique, le concept de variable, pourtant central, est quasiment inexistant et le concept de boucle semble être préféré à celui de structure conditionnelle chez les enfants. Plusieurs questions se posent concernant les concepts abordés par les dispositifs présentés. Constituent-ils la base à enseigner à un enfant ? Compte tenu qu'il existe peu d'études sur les difficultés rencontrées par ceux-ci dans l'apprentissage de la programmation, cela reste donc à prouver. On peut également se demander si un enfant perçoit réellement les concepts mis en œuvre dans l'utilisation d'un dispositif et s'il assimile ces concepts.
Cette classification est actuellement encore en cours de développement avec des résultats appelés à être enrichis. Des critères supplémentaires en didactique, en pédagogie notamment via les approches utilisées, mais également inspirés de la classification de Yu et Roque et orientés vers une description des blocs de programmation sont également envisagés.

Bibliographie

[1] Kunal Chawla, Megan Chiou, Alfredo Sandes, and Paulo Blikstein. 2013. Dr. Wagon: a'stretchable'toolkit for tangible computer programming. In *Proceedings of the 12th international conference on interaction design and children*. ACM, 561–564.

[2] Mollie Elkin, Amanda Sullivan, and Marina Umaschi Bers. 2016. Programming with the KIBO robotics kit in preschool classrooms. *Computers in the Schools* 33, 3

(2016), 169–186.

[3] Kenneth P Fishkin. 2004. A taxonomy for and analysis of tangible interfaces. *Personal and Ubiquitous Computing* 8, 5 (2004), 347–358.

[4] Daniel Gallardo, Carles F Julia, and Sergi Jorda. 2008. TurTan: A tangible programming language for creative exploration. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*. IEEE, 89–92.

[5] Paul Gross and Kris Powers. 2005. Evaluating assessments of novice programming environments. In *Proceedings of the first international workshop on Computing education research*. ACM, 99–110.

[6] Michael S Horn and Robert JK Jacob. 2006. Tangible programming in the classroom: a practical approach. In *CHI'06 extended abstracts on Human factors in computing systems*. ACM, 869–874.

[7] Michael S Horn and Robert JK Jacob. 2007a. Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction*. ACM, 159–162.

[8] Michael S Horn and Robert JK Jacob. 2007b. Tangible programming in the classroom with tern. In *CHI'07 extended abstracts on Human factors in computing systems*. ACM, 1965–1970.

[9] Michael S Horn, Erin Treacy Solovey, R Jordan Crouser, and Robert JK Jacob. 2009. Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 975–984.

[10] Michael S Horn, Erin Treacy Solovey, and Robert JK Jacob. 2008. Tangible programming and informal science learning: making TUIs work for museums. In *Proceedings of the 7th international conference on Interaction design and children*. ACM, 194–201.

[11] Felix Hu, Ariel Zekelman, Michael Horn, and Frances Judd. 2015. Strawbies: explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 410–413.

[12] Caitlin Kelleher and Randy Pausch. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)* 37, 2 (2005), 83–137.

[13] Vassilis Komis and Anastasia Misirli. 2011. Robotique pédagogique et concepts préliminaires de la programmation à l'école maternelle: une étude de cas basée sur le jouet programmable Bee-Bot. In *Sciences et technologies de l'information et de la communication en milieu éducatif: Analyse de pratiques et enjeux didactiques.* Athènes: New Technologies Editions, 271–281.

[14] Javier Marco, Clara Bonillo, and Eva Cerezo. 2017. A Tangible Interactive Space Odyssey to Support Children Learning of Computer Programming. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. ACM, 300–305.

[15] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

[16] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. 2007. A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin* 39, 4 (2007), 204–223.

[17] Radia Perlman. 1976. Using computer technology to provide a creative learning environment for preschool children. (1976).

[18] Martinha Piteira and Carlos Costa. 2012. Computer programming and novice programmers. In *Proceedings of the Workshop on Information Systems and Design of Communication*. ACM, 51–53.

[19] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.

[20] Anne Roudaut and Eric Lecolinet. 2007. Un espace de classification pour l'interaction sur dispositifs mobiles. In *Proceedings of the 19th Conference on l'Interaction Homme-Machine*. ACM, 99–106.

[21] Kimiko Ryokai, Michael Jongseon Lee, and Jonathan Micah Breitbart. 2009. Children's storytelling and programming with robotic characters. In *Proceedings of the seventh ACM conference on Creativity and cognition*. ACM, 19–28.

[22] Theodosios Sapounidis and Stavros Demetriadis. 2011. Touch your program with hands: qualities in tangible programming tools for novice. In *Informatics (PCI), 2011 15th Panhellenic Conference on*. IEEE, 363–367.

[23] Theodosios Sapounidis, Stavros Demetriadis, and Ioannis Stamelos. 2015. Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing* 19, 1 (2015), 225–237.

[24] Florian Scharf, Thomas Winkler, and Michael Herczeg. 2008. Tangicons: algorithmic reasoning in a collaborative game for children in kindergarten and first class. In *Proceedings of the 7th international conference on Interaction design and children*. ACM, 242–249.

[25] Orit Shaer, Eva Hornecker, and others. 2010. Tangible user interfaces: past, present, and future directions. *Foundations and Trends® in Human–Computer Interaction* 3, 1–2 (2010), 4–137.

[26] Arnan Sipitakiat and Nusarin Nusen. 2012. RoboBlocks: designing debugging abilities in a tangible programming system for early primary school children. In *Proceedings of the 11th International Conference on Interaction Design and Children*. ACM, 98–105.

[27] Andrew C Smith. 2010. Dialando: tangible programming for the novice with Scratch, processing and arduino. (2010).

[28] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 4 (2013), 15.

[29] Amanda Sullivan, Mollie Elkin, and Marina Umaschi Bers. 2015. KIBO robot demo: engaging young children in programming and engineering. In *Proceedings of the 14th international conference on interaction design and children*. ACM, 418–421.

[30] Hideyuki Suzuki and Hiroshi Kato. 1993. AlgoBlock: a tangible programming language, a tool for collaborative learning. In *Proceedings of 4th European Logo Conference*. 297–303.

[31] Hermans Felienne Swidan, Alaaeddin and Marileen Smit. 2018. Programming Misconceptions for School Students. In *Proceedings of the nternational Computing Education Research Conference: ICER '18*. ACM.

[32] Sureyya Tarkan, Vibha Sazawal, Allison Druin, Evan Golub, Elizabeth M Bonsignore, Greg Walsh, and Zeina Atrash. 2010. Toque: designing a cooking-based programming language for and with children. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2417–2426.

[33] Danli Wang, Yunfeng Qi, Yang Zhang, and Tingting Wang. 2013. TanPro-kit: a tangible programming tool for children. In *Proceedings of the 12th International Conference on Interaction Design and Children*. ACM, 344–347.

[34] Danli Wang, Tingting Wang, and Zhen Liu. 2014. A tangible programming tool for children to cultivate computational thinking. *The Scientific World Journal* 2014 (2014).

[35] Danli Wang, Cheng Zhang, and Hongan Wang. 2011. T-Maze: a tangible programming tool for children. In *Proceedings of the 10th international conference on interaction design and children*. ACM, 127–135.

[36] Danli Wang, Yang Zhang, and Shengyong Chen. 2013. E-Block: A tangible programming tool with graphical blocks. *Mathematical Problems in Engineering* 2013 (2013).

[37] Danli Wang, Yang Zhang, Tianyuan Gu, Liang He, and Hongan Wang. 2012. E-Block: a tangible programming tool for children. In *Adjunct proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 71–72.

[38] Peta Wyeth and Helen C Purchase. 2002. Tangible programming elements for young children. In *CHI'02 extended abstracts on Human factors in computing systems*. ACM, 774–775.

[39] Peta Wyeth and Gordon F Wyeth. 2001. Electronic blocks: Tangible programming elements for preschoolers. In *IFIP TC. 13 International Conference on Human-Computer Interaction*, Vol. 1. IOC Press, 496–503.

[40] Lesley Xie, Alissa N Antle, and Nima Motamedi. 2008. Are tangibles more fun?: comparing children's enjoyment and engagement using physical, graphical and tangible user interfaces. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*. ACM, 191–198.

[41] Junnan Yu and Ricarose Roque. 2018. A survey of computational kits for young children. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*. ACM, 289–299.